출석수업 과제물(평가결과물) 표지(온라인제출용)

**교과목명 : 알고리즘**

**학 번 : 202034-153746**

**성 명 : 이동열**

**강 의 실 : 인천지역대학 온라인**

**연 락 처 : 010-5264-5565**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1번)

분할정복 방법

* 이진탐색

탐색이란 배열에서 원하는 값을 찾는 문제이며, 이 중에서 이진 탐색은 배열이 정렬된 상태로 주어져 있을 때 효과적으로 찾는 방법을 말한다. 이진 탐색은 찾고자 하는 탐색키를 배열의 중앙 값과 비교한 후 같으면 멈추고 틀리면 해당 원소를 기준으로 부분 배열로 분리 해 다시 똑같은 방법으로 탐색을 반복하는 방식이다.

* 합병 정렬

합병 정렬은 한 배열을 같은 크기의 두 부분 배열로 나눈 후 순환적으로 합병 정렬을 수행 한 뒤 다시 합치는 정렬 방법을 말한다. 이 정렬 방식은 분할정복 방법에 해당하기 때문에 정렬 가능 한가장 작은 단위까지 나눈 후 다시 합병하는 방식이다.

* 퀵 정렬

피벗이라고 하는 특정한 원소를 정한 뒤 그 원소를 기준으로 배열을 나눈다. 그 다음 왼쪽 배열에는 더 피벗보다 작은 값을, 오른쪽에는 피벗보다 큰 값을 위치 시키는 방식이 퀵 정렬이다. 이것도 분할정복 방법이므로 분할된 배열들에 대해 순환적으로 퀵 정렬을 적용한다.

* 선택 문제

n개의 원소 중 k번째로 작은 원소를 구하는 문제로, 이 것을 구하는 첫 번째 방법은 배열을 오름차순으로 정렬 해 찾는 방법과 (k -1) 번 동안 최소 값을 찾아 삭제하는 동작을 반복하는 방법이 있다. 첫 번째 방법은 수행시간이 O(n log n)이고 두 번째 방법은 O(kn)이 된다.

동적 프로그래밍 방법

* 피보나치 수열 문제

위 점화식은 피보나치 수열을 정의 한 것으로, 피보나치 수열 자체가 위와 같이 점화식으로 정의 되어있기 때문에 최적성의 원리가 성립되어 동적 프로그래밍 방법을 적용 할 수 있는 문제이다.

* 연쇄 행렬 곱셈 문제

n개의 행렬을 곱할 때 이 행렬의 곱셈은 결합법칙이 성립 되기 때문에 어떤 순서로 곱셈을 수행하더라도 결과는 같다. 하지만 곱셈 순서에 따라 곱셈 횟수는 달라지는데 이때 최소의 곱셈 횟수를 만족하도록 곱셈 순서를 구하는 문제를 말한다.

* 스트링 편집 거리 문제

두 문자열이 존재 할 때 한 문자열을 다른 문자열로 변환하는데 필요한 최소 비용을 구하는 문제이다. 이 문제는 두 문자열 간 유사성을 판단하는 기준으로 활용 되는데 예를 들어 한 문자열이 철자 검사기에서 잘못된 결과가 나오면 가장 유사한 단어를 찾아주는 방식이다.

* 모든 정점 간의 최단 경로

가중 방향 그래프가 존재 할 때, 모든 정점 사이의 이동 조합 중에서 두 정점 사이의 가중치의 합이 가중치의 합이 제일 작은 경로를 구하는 문제

* 저울 문제

양팔 저울과 n개의 추가 있고 각 추마다 무게가 정해져 있을 때, 특정 무게를 조합해낼 수 있는지를 확인하는 문제이다. 만약에 가지고 있는 추를 조합해서 특정 무게와 같게 만들 수 있다면 이 문제는 해가 존재하는 것이다.

욕심쟁이 방법

* 동전 거스름돈 문제

물건을 사고 동전을 거슬러 준다고 할 때, 어떻게 해야 동전 개수를 최소화 해서 거슬러 줄 수 있는지 계산하는 문제이다. 이 문제는 금액이 가장 큰 동전부터 사용해서 거슬러 주는 방식으로 해결하며 성능은 n개의 동전 종류가 있을 경우 O(n)이다.

* 배낭 문제

용량이 정해져 있는 배낭과 가치가 정해져 있는 물건들이 존재 할 때, 가장 큰 이익을 낼 수 있도록 배낭에 물건을 채우는 문제이다. 물체는 나눠서 넣을 수 있으며 단위 무게당 가치가 큰 물건들을 먼저 넣는 방식으로 해결 할 수 있다. 성능은 물건이 n개가 존재 할 경우 물건이 가치에 맞게 정렬되어 있으면 O(n), 정렬되어 있지 않으면 O(n log n)이다.

* 최소 신장 트리

가중 무방향 그래프가 존재 할 때, 신장 트리에서 최소 가중치의 합을 구하는 문제이다. 크루스칼 알고리즘과 프림 알고리즘을 사용해 구할 수 있다.

* 최단 경로

가중 그래프가 존재 할 때, 두 정점 사이의 모든 경로 중 최단 경로를 구하는 문제이다. Dijkstra 알고리즘을 사용해 해결 할 수 있다.

* 작업 스케줄링 문제

여러 개의 작업이 존재 할 때, 최소한의 기계만 사용해 작업 간 충돌없이 작업을 분배하는 문제이다. 이 문제는 시작 시간이 빠른 작업을 순서로 충돌 여부를 확인 후 기존 기계나 새 기계에 할당 하는 방식으로 해결 할 수 있다. 성능은 작업이 n 개 있다고 할 경우 O(n log n)이다.

* 작업 선택 문제

한 개의 기계로 충돌 없이 최대한 많은 개수의 작업을 할당하는 문제이다. 작업 시간이 짧은 작업을 우선적으로 선택해 충돌 여부 확인 후 할당하는 방식이며, 작업이 n개가 존재 할 경우 O(n log n) 성능이 나온다.

* 허프만 코딩

텍스트에서 문자가 나타나는 빈도수에 따라 코드를 부여해 전체 텍스트를 줄이는 문제이다. 자주 나타나는 문자는 짧은 코드를 부여하고 자주 나타나지 않은 문자는 긴 코드를 부여하는 방식이다.

2번

3번

* 이진 탐색
* 퀵 정렬의 최악의 경우
* 합병 정렬
* 퀵 정렬의 최선의 경우

4번

partition() 함수는 배열과 범위를 인자로 받아 피벗을 기준으로 두 배열로 분할 해 왼쪽 배열에는 피벗보다 작은 값을, 오른 쪽에는 피벗보다 높은 값을 위치 시키는 함수이다.

A[] = { 35 50 25 40 70 20 45 55 30 10} 와 같은 함수가 있다 하면,

partition([35,50,25,40,70,20,45,55,30,10], 10) 이런 식으로 호출 할 수 있다. 두 번째 인자로 전달 한 값은 정렬시킬 원소의 개수이다. 10을 전달했고 10은 배열의 크기와 동일하기 때문에 배열 전체를 분할한다.

이때 A[0](35)을 피벗으로 해서 분할하면

[25, 20, 30, 10], 35, [50, 40, 70, 45, 55]로 분할 할 수 있다.

이 상태에서 다시 합치면

[25, 20, 30, 10, 35, 50, 40, 70, 45, 55]가 되고 partition() 함수는 피벗의 인덱스인 4를 반환하게 된다.

5번

A[] = { 35 50 25 40 70 20 45 55 }

먼저 배열의 중간 위치를 기준으로 배열을 분할한다.

[35, 50, 25, 40] , [70, 20, 45, 55] 이렇게 두 배열이 나오면 왼쪽 배열에 다시 합병 정렬을 적용한다.(순환 적용)

[35, 50], [25, 40] 여기서 또 왼쪽 배열에 합병 정렬을 적용 한다.

[35], [50] 더 이상 정렬이 불가능 하므로 합병 시킨다.

[35, 50], [25, 40] 여기서 이번엔 오른쪽 배열에 합병 정렬을 적용 시킨다.

[25], [40] 더 이상 정렬이 불가능 하므로 합병 시킨다.

[35, 50], [25, 40] 이 두 배열은 정렬 되어 있는 상태이므로 합병한다.

[25, 35, 40, 50], [70, 20, 45, 55] 이제 오른쪽 배열에 같은 방법으로 합병 정렬을 적용한다.

[70, 20], [45, 55] 왼쪽 배열에 합병 정렬 적용

[70], [20] 더 이상 정렬 불가능 하므로 합병

[20, 70], [45, 55] 오른쪽 배열에 합병 정렬 적용

[45], [55] 더 이상 정렬 불가능 하므로 합병

[20, 70], [45, 55] 두 배열 모두 정렬되었으므로 합병

[25,35,40,50],[20, 45, 55 ,70] 두 배열 모두 정렬되었으므로 합병

[20, 25, 35, 40, 45, 50, 55, 70]

6번

(1)

A[] = { 35 50 25 40 70 20 45 55 } 가 존재 할 때,

1.먼저 A[0]을 루트 노트로 삽입한다.

2.A[1]을 루트 노드의 왼쪽 자식 노드로 삽입한다.

3. 2에서 삽입한 자식 노드가 부모인 루트 노드보다 값이 크므로 둘의 위치를 변경한다.

4. A[2]를 루트 노드의 오른쪽 자식 노드로 삽입 한다.

5. 4에서 삽입한 노드 25가 부모인 루트 노드 보다 값이 작으므로 위치를 유지한다.

6. A[3]을 노드 35의 왼쪽 자식 노드로 삽입한다.

7. 6에서 삽입한 노드 40이 부모 노드인 35보다 값이 크므로 둘의 위치를 변경한다.

8. A[4]를 노드 40의 오른쪽 자식 노드로 삽입한다.

9. 8에서 삽입한 노드 70이 부모 노드인 40보다 값이 크므로 둘의 위치를 변경한다.

10. 위치가 변경된 노드 70이 부모 노드인 50보다 값이 크므로 둘의 위치를 변경한다.

11. A[5]를 노드 25 왼쪽 자식 노드로 삽입한다.

12. 11에서 삽입한 노드 20이 부모 노드인 25보다 값이 작으므로 위치를 유지한다.

13. A[6]을 노드 25 오른쪽 자식 노드로 삽입한다.

14. 13에서 삽입한 노드 45가 부모 노드인 25보다 값이 크므로 위치를 변경한다.

15. A[7]을 노드 35의 왼쪽 자식 노드로 삽입한다.

16. 15에서 삽입한 노드 55가 부모 노드인 35보다 값이 크므로 위치를 변경한다.

17. 노드 55가 부모 노드인 50보다 값이 크므로 둘의 위치를 변경한다. (완료)

(2)

힙 정렬 첫 번째 수행

1. 초기 힙에서 최대 값인 루트 노드 70과 배열 상에서 제일 오른쪽 노드인 노드 35의 위치를 변경한다.
2. 최대 값을 삭제 한다.
3. 노드 35가 왼쪽 자식 노드 55보다 값이 작으므로 위치를 변경한다.
4. 노드 35가 왼쪽 자식 노드 50보다 값이 작으므로 둘의 위치를 변경한다.

힙 정렬 두 번째 수행

1. 최대 값인 루트 노드 55와 배열 상 제일 오른쪽 노드인 노드 25의 위치를 변경한다.

1. 노드 55를 삭제한다.
2. 노드 25보다 왼쪽 자식 노드인 50의 값이 더 크므로 둘의 위치를 변경한다.
3. 노드 25보다 오른쪽 자식 노드인 40의 값이 더 크므로 둘의 위치를 변경한다.